Announcements

- **Sunday, November 1\(^{st}\) at 11:59pm:**
  - Homework Project 1 late deadline
  - 25% penalty on score
- **Sunday, November 8\(^{th}\) at 11:59pm:**
  - Homework Project 2 due
Lecture Overview

- Texture Mapping
  - Overview
  - Wrapping
  - Texture coordinates
Large Triangles

Pros:
- Often sufficient for simple geometry
- Fast to render

Cons:
- Per vertex colors look boring and computer-generated
Texture Mapping

- Map textures (images) onto surface polygons
- Same triangle count, much more realistic appearance
Texture Mapping

- **Goal**: map locations in texture to locations on 3D geometry

- **Texture coordinate space**
  - Texture pixels (texels) have texture coordinates \((s, t)\)

- **Convention**
  - Bottom left corner of texture is at \((s, t) = (0, 0)\)
  - Top right corner is at \((s, t) = (1, 1)\)
Texture Mapping

- Store 2D texture coordinates \(s, t\) with each triangle vertex

\[
\begin{align*}
\mathbf{v}_1 & \quad (s, t) = (0.65, 0.75) \\
\mathbf{v}_0 & \quad (s, t) = (0.6, 0.4) \\
\mathbf{v}_2 & \quad (s, t) = (0.4, 0.45)
\end{align*}
\]

Triangle in any space before projection

Texture coordinates
Texture Mapping

- Each point on triangle gets color from its corresponding point in texture

\[ \mathbf{v}_1 \]
\[ (s,t) = (0.65,0.75) \]

\[ \mathbf{v}_0 \]
\[ (s,t) = (0.6,0.4) \]

\[ \mathbf{v}_2 \]
\[ (s,t) = (0.4,0.45) \]

Triangle in any space before projection

Texture coordinates
Texture Mapping

Primitives

- Modeling and viewing transformation
  - Shading
  - Projection
    - Rasterization
      - Fragment processing
        - Frame-buffer access (z-buffering)
          - Frame-buffer access
            - Includes texture mapping
          - Image
Texture Look-Up

- Given interpolated texture coordinates \((s, t)\) at current pixel
- Closest four texels in texture space are at 
  \((s_0, t_0), (s_1, t_0), (s_0, t_1), (s_1, t_1)\)
- How to compute pixel color?
Nearest-Neighbor Interpolation

- Use color of closest texel

- Simple, but low quality and aliasing
Bilinear Interpolation

1. Linear interpolation horizontally:

   Ratio in s direction $r_s$:

   $\frac{s - s_0}{s_1 - s_0}$

   $c_{\text{top}} = \text{tex}(s_0, t_1) (1 - r_s) + \text{tex}(s_1, t_1) r_s$

   $c_{\text{bot}} = \text{tex}(s_0, t_0) (1 - r_s) + \text{tex}(s_1, t_0) r_s$
Bilinear Interpolation

2. Linear interpolation vertically

Ratio in t direction \( r_t \):

\[
 r_t = \frac{t - t_0}{t_1 - t_0}
\]

\[
 c = c_{\text{bot}} (1 - r_t) + c_{\text{top}} r_t
\]
Texture Filtering in OpenGL

- **GL_NEAREST**: Nearest-Neighbor interpolation
- **GL_LINEAR**: Bilinear interpolation

**Example:**

- `glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);`
- `glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);`

Source: https://open.gl/textures
Lecture Overview

- Texture Mapping
  - Overview
  - Wrapping
  - Texture coordinates
Wrap Modes

- Texture image extends from [0,0] to [1,1] in texture space
  - What if \((s,t)\) texture coordinates are beyond that range?
  - → Texture wrap modes
Repeat

- Repeat the texture
  - Creates discontinuities at edges
    - unless texture is designed to line up

Texture Space

Seamless brick wall texture
(by Christopher Revoir)
Clamp

- Use edge value everywhere outside data range \([0..1]\)
- Or use specified border color outside of range \([0..1]\)
Wrap Modes in OpenGL

- **Default:**
  - `glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT );`
  - `glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT );`

- **Options for wrap mode:**
  - `GL_REPEAT`
  - `GL_MIRRORED_REPEAT`
  - `GL_CLAMP_TO_EDGE`: repeats last pixel in the texture
  - `GL_CLAMP_TO_BORDER`: requires border color to be set

Source: [https://open.gl/textures](https://open.gl/textures)
OpenGL Example: Loading a Texture

// Loads image as texture, returns ID of texture
GLuint loadTexture(Image* image)
{
    GLuint textureId;

    glGenTextures(1, &textureId); // Get unique ID for texture
    glBindTexture(GL_TEXTURE_2D, textureId); // Tell OpenGL which texture to edit
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR); // set bi-linear interpolation
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR); // for both filtering modes
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE); // set texture edge mode
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);

    Image* image = loadJPG("photo.jpg"); // load image from disk; uses third party Image library

    // Depending on the image library, the texture image may have to be flipped vertically

    // Load image into OpenGL texture in GPU memory:
    glTexImage2D(GL_TEXTURE_2D, // Always GL_TEXTURE_2D for image textures
                 0, // 0 for now
                 GL_RGB, // Format OpenGL uses for image without alpha channel
                 image->width, image->height, // Width and height
                 GL_RGB, // GL_RGB, because pixels are stored in RGB format
                 GL_UNSIGNED_BYTE, // GL_UNSIGNED_BYTE, because pixels are stored as unsigned numbers
                 image->pixels); // The actual RGB image data

    return textureId; // Return the ID of the texture
}
# Vertex Shader

```glsl
#version 150

in vec3 vert;
in vec2 vertTexCoord;
out vec2 fragTexCoord;

void main()
{
    // Pass the tex coord straight through to the fragment shader
    fragTexCoord = vertTexCoord;

    gl_Position = vec4(vert, 1);
}
```
# Fragment Shader

```glsl
#version 150

uniform sampler2D tex; // this is the texture
in vec2 fragTexCoord; // these are the texture coordinates
out vec4 finalColor;   // this is the output color of the pixel

void main() {
    finalColor = texture(tex, fragTexCoord);
}
```

22
Lecture Overview

- Texture Mapping
  - Overview
  - Wrapping
  - Texture coordinates
Texture Coordinates

What if texture extends across multiple polygons?

→ Surface parameterization

- Mapping between 3D positions on surface and 2D texture coordinates
  - Defined by texture coordinates of triangle vertices

- Options for mapping:
  - Cylindrical
  - Spherical
  - Orthographic
  - Parametric
  - Skin
Cylindrical Mapping

- Similar to spherical mapping, but with cylindrical coordinates
Spherical Mapping

- Use spherical coordinates
- “Shrink-wrap” sphere to object

Texture map

Mapping result
Orthographic Mapping

- Use linear transformation of object’s xyz coordinates
- Example:

\[
\begin{bmatrix}
  s \\
  t \\
\end{bmatrix} = \begin{bmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
\end{bmatrix} \begin{bmatrix}
  x \\
  y \\
  z \\
  w \\
\end{bmatrix}
\]

---

```
x in object space   x in camera space
```
Parametric Mapping

- Surface given by parametric functions
  \[ x = f(u, v) \quad y = f(u, v) \quad z = f(u, v) \]
- Very common in CAD
- Clamp \((u,v)\) parameters to \([0..1]\) and use as texture coordinates \((s,t)\)
Skin Mapping